Offline GA-Based Optimization for Heterogeneous Modular Multiconfigurable Chained Microrobots

Alberto Brunete Miguel Hernando Ernesto Gambao

Abstract—**This paper presents a GA-based optimization procedure for bioinspired heterogeneous modular multiconfigurable chained microrobots. When constructing heterogeneous chained modular robots that are composed of several different drive modules, one must select the type and position of the modules that form the chain. One must also develop new locomotion gaits that combine the different drive modules. These are two new features of heterogeneous modular robots that they do not share with homogeneous modular robots. This paper presents an offline control system that allows the development of new configuration schemes and locomotion gaits for these heterogeneous modular multiconfigurable chained microrobots. The offline control system is based on a simulator that is specifically designed for chained modular robots and allows them to develop and learn new locomotion patterns.**

I. Introduction

THE development of new locomotion patterns in bioinspired modular robots is one of the most challenging issues in this field. Significant research has been performed in homogeneous modular robots [1]–[6] and their different locomotion modes [7]–[10]. This paper presents the development of locomotion modes for heterogeneous modular robots by using genetic algorithms (GAs).

GAs can be found in many homogeneous modular robots and are used to develop their controllers or algorithms. In M-TRAN [11], an automatic locomotion generation method (ALPG) is used to produce locomotion in arbitrary module configurations using a neural oscillator as a model for the central pattern generator (CPG) and GAs for evolving parameters. ATRON robot [12] uses GAs to determine how well the combination of modules performs a given task when artificial evolution has been used to develop the individual controllers. The evolutionary algorithm was a simple GA working on a string of bytes. YaMoR [13] presents a GA that coevolves the CPGs with the configuration of the modular robot to achieve the rapid online optimization (i.e., adaptation) of the locomotion gaits in response to module failures or new, previously unknown configurations. Recent papers like [14]–[16] also present GAs for the optimization of tasks and configurations of modular homogeneous reconfigurable robots and manipulators.

In recent years, the importance of heterogenous modular microrobots has increased, as stated in [17]: a complex system behavior can be obtained by the integration of a large number of simple different micromodules, each having a specific simple task (power, locomotion, sensing, manipulation, communication, or computation). However, not much research has considered heterogeneous modular robots. This is problematic because heterogeneous chained modular robots are composed of several drive modules. It is important to determine the most appropriate locomotion patterns of the combination of the different drive modules and the importance of the position of the modules in the chain (which is irrelevant when the modules are homogeneous). One of the few heterogeneous modular robots that has been developed thus far is the one by Jantapremjit and Austin [18]. However, that project was abandoned due to its difficulty and cost.

The authors have previously proposed a control architecture for a bioinspired semiautonomous multiconfigurable heterogeneous microrobot for the inspection and maintenance of small-diameter pipes and cavities. The microrobot, called Microtub [19], was designed as a means to explore pipes with a camera to detect breakages, holes, leaks, and other defects. This microrobot is composed of different modules, each of which performs a different movement or task. Thus, multiconfigurability is an essential characteristic of the microrobot: one should be able to easily interchange these modules, depending on the required task, without reprogramming the microrobot.

This paper presents an offline GA-based optimization algorithm for computing the optimal modular configuration of the robot and its locomotion parameters in different situations. Given several heterogeneous modules that can be chained together, the algorithm determines the optimal position in the chain for each of them and identifies the optimal locomotion parameters (sinusoidal frequency and amplitude, extension and contraction times, etc.) for the given configuration. Offline control optimization permits the development of rules and patterns subsequently used in the online control.

The algorithm proposed in this paper can be easily generalised to any bioinspired chained modular robot with the condition that the new modules must be simulated. Subsequently, the algorithm will be ready to be run on the new robot.

This paper is structured as follows. Section II presents the robot and the previous work. Section III presents the offline control algorithms, and Section IV discusses several experiments

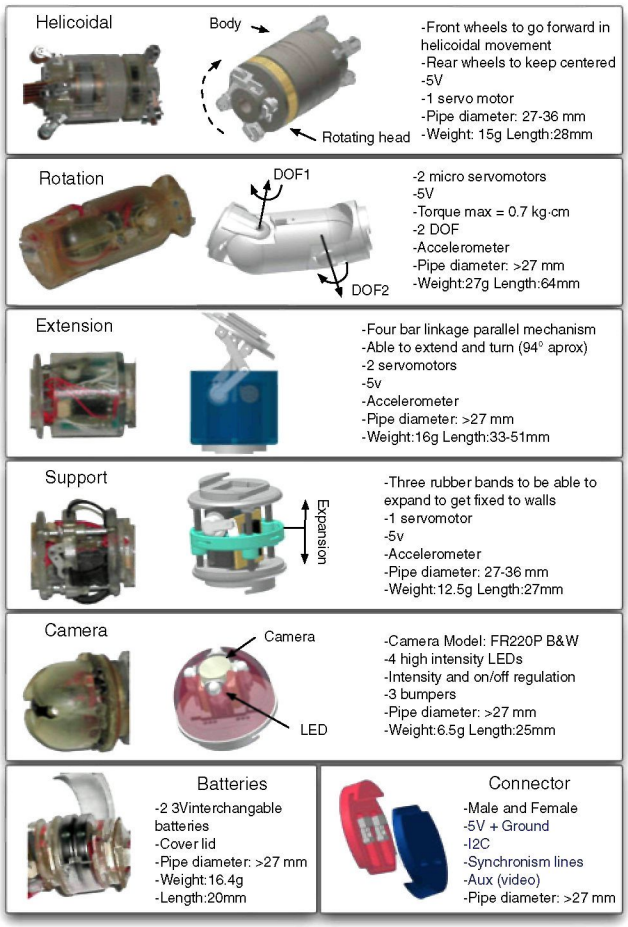

Fig. 1. Module description chart.

that have been performed to validate the proposed offline control.

II. Previous Work and Setup

To understand the offline optimization algorithms that are presented here, one must understand the control architecture and the robot for which it has been developed. This section briefly presents the robot and its control architecture.

A. Robot Description

As previously mentioned, the Microtub microrobot is heterogeneous and modular, meaning that it is composed of different types of active modules (ones that are able to move) and passive modules (ones that have to be acted on).

The different types of modules that have been developed can be observed in Figs. 1 and 2. More information can be found in [20]. The diameter of each module is only 27 mm and the thickness of some parts is less than 1 mm.

The camera/contact module is used for environment information acquisition as necessary to detect holes, breakages, or cracks in the pipes. It is also used as a contact sensor to detect if the microrobot is facing an obstacle. The rotation module is

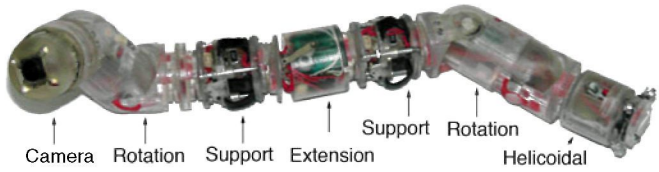

Fig. 2. Modules assembled together.

a 2 [DOF] module that allows rotation on the horizontal and vertical planes. A set of these modules put together can perform an undulating (snake-like) movement. The support and extension modules are used to perform inchworm movements (move forward and turn right and left). Finally, the helicoidal module was designed to be a fast-drive module that is able to push other modules.

To assemble the modules together, a common interface has been built. This interface allows for mechanical and electrical connections between the modules.

Each module includes an electronic control board that performs the following tasks: control of actuators and sensors, communications, autoprotection and adaptable motion, self-orientation detection, and low-level embedded control.

B. Simulator

A simulation environment has been developed for effective prototype testing and to validate the control algorithms, hardware design, and system deployment scenarios. It can also be used to verify the feasibility of system behaviors using realistic morphology, body mass, and torque specifications for servos.

The simulator is built on the Open Dynamics Engine (ODE), which is an open-source physics simulation API that makes it possible to perform online simulations of rigid body dynamics and to examine a wide variety of experimental environments and models. Based on the ODE, an entire simulation system has been developed from scratch that includes the mechanical features of the modules, DOF, movement ranges, servomotors, communications, and processing units.

In addition to simulating the dynamic behavior of the robot, the simulator can be used to emulate the robots electronics, control, communications, and processing system. Thus, protective behaviors, such as the prevention of motor overheating, can be tested. In the same way, each simulated microcontroller has its own thread assigned in the simulator, which has facilitated the implementation of synchronization mechanisms using the different modules.

The simulator has been previously validated in several experiments, including servomotor, friction, speed, and locomotion tests. Fig. 3 shows a comparison of the intensity and torque of the simulated servo motor and the real one, and Table I shows a comparison of the speeds attained by the robot performing an inchworm gait in simulated and real environments. The gap between real and simulated data is due to the difficulty of simulating accurately the rubber strips; therefore, parameters have been optimized for the horizontal position.

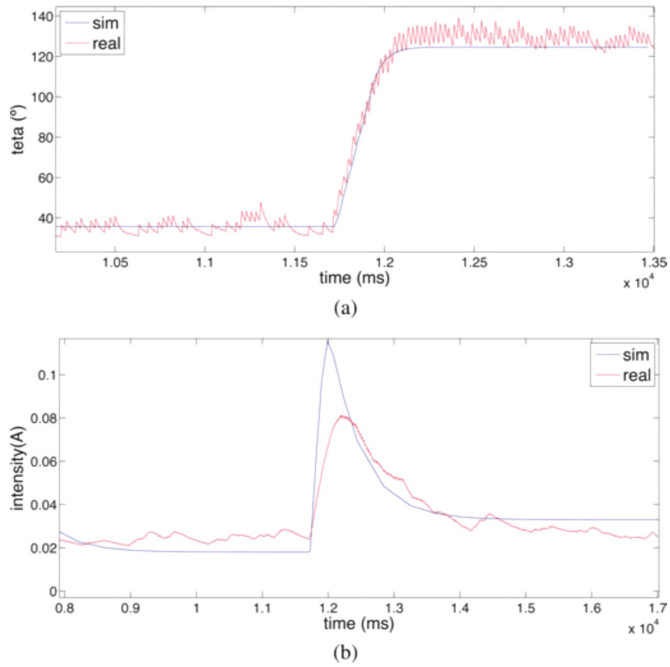

Fig. 3. Simulator validation tests. (a) Rotation angle. (b) Current intensity.

TABLE I
SPEED TEST OF THE INCHWORM CONFIGURATION

Angle (°)	0	30	90
Speed (cm/s) (Real)	2,5	1,5	0,6
Speed (cm/s) (Simulation)	2,4	1,3	0,3

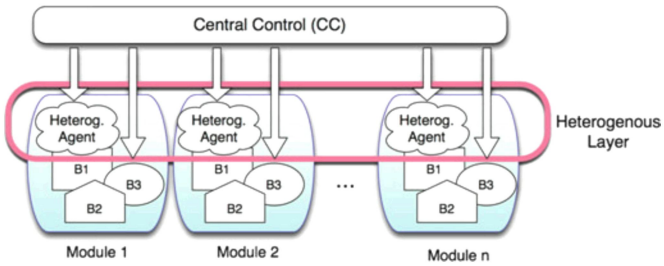

Fig. 4. Control layers.

C. Control Architecture

Microtub features a semidistributed control with a central control (CC) that makes decisions for the whole robot and an embedded behavior-based control in every module that can react in real time to unpredicted events. There is also an interpreter in each module that interfaces between the CC and the behaviors. This is called the *heterogeneous agent*. The heterogeneous agents of all of the modules form the heterogeneous layer, which is called a middle layer because it connects the CC (the highest layer) and the onboard control (the lowest layer). Thus, the control is divided into a CC and an embedded control (see Fig. 4).

The CC is currently placed in a PC, but it could be positioned in one of the modules to make the robot autonomous. It includes the high-control layer, which controls the robot as a whole. It collects information from the modules, processes it, and subsequently transmits information about the situation and state of the robot. It also sends commands with objectives to the modules and helps the modules to make and coordinate decisions. In addition, the high-control layer is in charge of planning and is composed of several parts, including an inference engine and a behavior-based control.

The onboard control is embedded in each module and is based on behaviors. It includes the following layers.

1) The heterogeneous (middle) layer translates the commands from the CC into specific module commands. For example, it translates the command "extend" into servomotor movements.
2) The low-control layer is composed of behaviors. It allows the modules to react in real time (for example, to sense external and internal stimuli, such as overheating and unreachable positions, and to adapt to the shape of the pipe) and to perform tasks that do not require the CC (movements, communication with adjacent modules, etc.).

III. OFFLINE CONTROL

Offline control involves the control algorithms that are used when the microrobot is not running. These algorithms select the best configuration for the microrobot (in terms of both module positioning and parameters) for later use in the online control mode.

Offline optimization permits the development of rules and movement patterns based on a simulator that is specifically designed for this type of robot.

Two options for GA are considered.

1) *Parameter optimization*: For a given configuration, the GA must determine the optimum parameters for ideal performance (amplitude, phase, or frequency in snake-like movements, extension/contraction time during inchworm movements). This option is especially useful in homogeneous configurations when the microrobot is performing a snake-like or inchworm movement.
2) *Configuration demand*: In heterogeneous configurations, the GA has to determine the modules to use to determine the best combination for performing a specific task: to cover a stretch of pipe, to negotiate an elbow, or to achieve the lowest amount of power consumption.

Due to the differing nature of the two options considered, the way to resolve each option will be revealed in the implementation of the algorithm.

A. Codification and Setup

In this phase, the parameters used in the GA must be defined, including the chromosomes, population, number of generations, fitness function, and termination condition. The chromosome is one of the most important parameters to define. If the chromosome is not well chosen, good results will be impossible to achieve.

1) Configuration Demand: The chromosome is an array of the types of modules of the robot (i.e., genes). If the robot has six modules, the chromosome will be an array of six elements, each of them representing the type of module (rotation, helicoidal,

TABLE II
GA CONFIGURATION DEMAND GENES VALUE RANGE

Parameter	Value
Rotation module	1
Helicoidal module	2
Support module	3
Extension module	4
Touch/Camera module	5
Traveler module	6

TABLE III
GA PARAMETER OPTIMIZATION GENES VALUE RANGE

Parameter	Value
Amplitude	$[-90°..90°]$
Angular velocity	$[0..15]$
Phase	$[0..\pi]$
Offset	$[0..90°]$
Phase between vertical and horizontal modules	$[0..2\pi]$

support, extension, touch/camera, or traveller) according to Table II.

For example, for a microrobot composed of 1 touch module, 5 rotation modules and 1 helicoidal module, the chromosome is "5111112." For a microrobot composed of 1 touch, 1 support, 1 extension, 1 support, 1 rotation, 1 support, 1 extension, 1 support, and 1 rotation module, the chromosome is "534313431."

The population is the set of chromosomes, which varies between 20 and 200 depending on the experiment.

The fitness function may vary. It can be related to the time that the microrobot takes to perform a task (i.e., to cover a part of the pipe the shorter the time is, the better the fitness value is) or to the distance covered in a particular amount of time (the larger, the better).

The probabilities that are experimentally chosen for this algorithm are as follows.

1) Crossover probability = 0.7–0.8.
2) Mutation probability = 0.005–0.01.

2) Parameter Optimization: The chromosome is an array of parameters that define the movement of the microrobot. If the robot is composed of rotation modules that are used to perform a snake-like movement, this parameter can be amplitude A, angular velocity V, phase P, offset O, and phase between vertical and horizontal modules D. The range of the values for these parameters can be observed in Table III. The chromosome is the array "$AVPOD$." For example: "60.0; 1.0; 2π/3; 0; 0".

Because each parameter has a different range, its values must be converted to a common range. The range that has been selected is [0..127]. This value is converted into binary code (seven digits for each value, 2^7) and is ready to be used.

The previous example becomes "42; 8; 84; 0; 0," but now, all values are within [0..127], which in binary is

"0101010; 0001000; 1010100; 0000000; 0000000"

Thus, the chromosome is:

"01010100001000101010000000000000000."

If the robot is composed of support and extension modules that are used to perform an inchworm movement, this parameter can be extension time T, expansion time P, extension length L, and support servo angle S. The chromosome is "$TPLS$," and it will exhibit the same transformation as before.

B. Phases of the GAs

1) Initialization: Initially, many individual solutions are randomly generated to form the initial population that was previously specified. The population size depends on the nature of the problem, but the population typically contains several hundreds or thousands of possible solutions. The population is generated randomly and covers the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

In the configuration demand, the genes are random numbers between 0 and 5. In the parameter optimization, the genes are random bits that correspond to random numbers that are selected from the possible values, as observed in Table III.

2) Evaluation: The evaluation involves applying the fitness function to every chromosome within the population.

The fitness function evaluates the performance of the chromosome. It transforms the chromosome into the modules and parameters represented by the chromosome and runs the simulator with these modules.

The fitness function follows these steps:

1) starts the simulation;
2) creates the modules specified by the genes (configuration demand) or applies the parameters to the modules (parameter optimization);
3) runs the simulation (more rapidly than a normal simulation) to achieve the specified objective;
4) terminates the simulation when either the objectives are completed or the maximum number of iterations has been reached;
5) returns a value: time[s], if the goal is to cover a part of the pipe, to cover a distance in open air or negotiate an elbow in as little time as possible, or intensity[A] if the goal is to minimized the power consumption.

The values returned by the fitness function are stored for later use in the selection phase.

For example, let us suppose that we have six chromosomes composed of six modules:

C1 : RRRRRR = 111111
C2 : CRRRRH = 511112
C3 : RRRHHH = 111222
C4 : CRSEST = 513436
C5 : SESSES = 343343
C6 : RSRSRS = 131313

and the fitness function calculates the distance covered in a straight pipe in 20 s. The following results are obtained:

C1 : 0.4 m
C2 : 0.6 m
C3 : 1 m
C4 : 0.45 m
C5 : 0.8 m
C6 : 0.2 m

3) Selection: After the entire population has been evaluated, the selection phase begins. During each generation, the part of

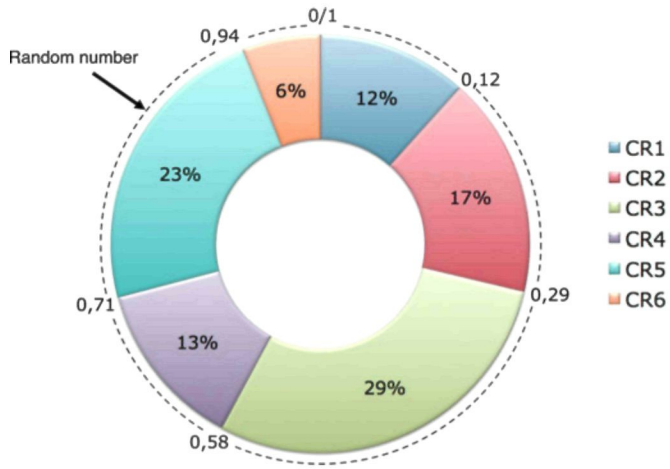

Fig. 5. Roulette probability.

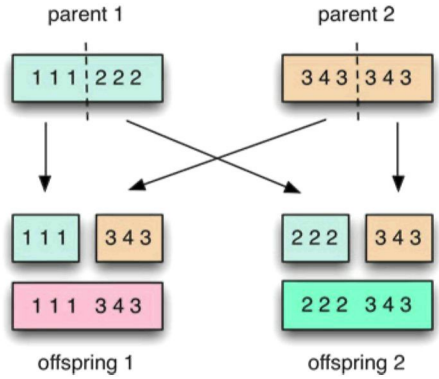

Fig. 6. Single-point crossover example.

the existing population selected to breed a new generation is called the mating pool. Individual solutions are selected using a fitness-based process in which fitter solutions are more likely to be selected.

The basic selection process involves stochastically selecting from one generation to create the basis of the next generation. The requirement for this process is that the fittest chromosomes must have a greater chance of transmitting their genetic information than the weaker ones. This characteristic of the process replicates nature in that fitter individuals will tend to have a better probability of survival and will become part of the mating pool for the next generation. Weaker individuals, however, still have a chance to become part of the mating pool. In nature, such individuals may have genetic coding that may prove useful to future generations.

In this paper, the roulette wheel selection method, which employs stochastic sampling with replacement, has been used.

This sampling method selects parents according to the spinning of a weighted roulette wheel (see Fig. 5). The roulette wheel is weighted according to the fitness values that have been previously obtained. A high fit value will have more area assigned to it on the wheel and hence will have a higher probability of being selected when the biased roulette wheel is spun. Roulette wheel selection is a high-variance process that features a fair amount of scatter between the expected and actual number of copies.

Taking the example of the distance covered in 20 s, we can see that the previous chromosomes have the following selection probabilities.

$C1 : 0.12, 12\%$
$C2 : 0.17, 17\%$
$C3 : 0.29, 29\%$
$C4 : 0.13, 13\%$
$C5 : 0.23, 23\%$
$C6 : 0.06, 6\%$

The probability range for each chromosomes is as follows:

$C1 : \text{From } 0 \text{ to } 0.12\ [0, 0.12]$
$C2 : \text{From } 0.12 \text{ to } 0.29,\ (0.12,\ 0.29]$
$C3 : \text{From } 0.29 \text{ to } 0.58,\ (0.29, 0.58]$
$C4 : \text{From } 0.58 \text{ to } 0.71,\ (0.58, 0.71]$
$C5 : \text{From } 0.71 \text{ to } 0.94,\ (0.71, 0.94]$
$C6 : \text{From } 0.94 \text{ to } 1,\ (0.94, 1]$

We obtain the following random numbers: 0.08, 0.4, 0.68, 0.45, 0.015, and 0.9. C3 is selected twice, C6 zero times, and the remaining chromosomes are selected once each.

4) Reproduction: The next step is to generate a second-generation population of solutions from those selected in the "selection" phase using genetic operators: crossover (also called recombination) and/or mutation.

For each new solution to be produced, a pair of "parent" solutions will be selected for breeding from the previously selected mating pool. By producing a "child" solution using the aforementioned methods of crossover and mutation, a new solution is created that typically exhibits many of the characteristics of its "parents." New parents are selected for each new child, and the process continues until a new population of solutions of the appropriate size is generated.

These processes ultimately result in the next-generation population of chromosomes, which is different from that of the initial generation. Generally, the average fitness of the population will increase via this procedure because only the best organisms from the first generation are selected for breeding along with a small proportion of less fit solutions (for the reasons mentioned previously).

In this paper, "single-point crossover" is used (see Fig. 6): one crossover point is selected, the genes from the beginning of the chromosome to the crossover point are copied from one parent, and the rest are copied from the second parent. The "single-point crossover" method has been experimentally chosen because it provides a good tradeoff between exploration (the introduction of new features) and exploitation (the retention of good features). The crossover point is selected randomly and could be any number between 1 and the length of the chromosome minus 1.

Then, mutation is performed. Each gene of each chromosome may be changed based on the probability of the mutation. For each gene, a random number is selected, and if it is smaller than the mutation probability, the gene is switched with another number that is obtained randomly (see Fig. 7). The point of crossover and the mutation probabilities are pseudorandom variables following a uniform distribution.

Fig. 7. Mutation example.

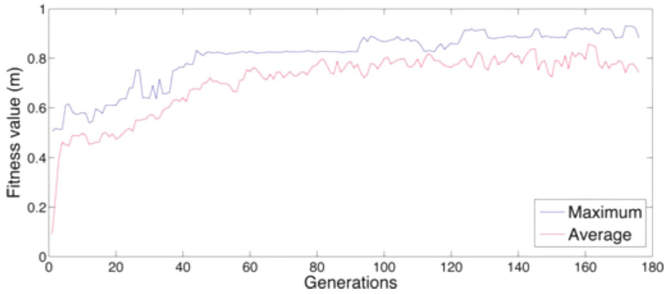

Fig. 8. Results of the GA in open air (parameter optimization).

In the same example as before, we take two parents for the crossover, "111222" and "343343." Then, a number from 1 to 5 is selected randomly, and 3 is obtained. Thus, the first three genes of parent 1 go to the first offspring, and the last three genes go to the second offspring. The opposite occurs with parent 2.

In the case of mutation, if the chosen parent is "513436," a random number from 0 to 1 is obtained for each gene. If the number is smaller than the mutation probability, then the gene changes. In this case, the only gene that changes is the fourth one. Another random number is selected from 1 to 6 (the number of modules) to replace the gene. In this case, the new gene is "1."

5) Termination: This generational process is repeated until a termination condition has been reached.

In our case, a fixed number of generations must be reached, the highest ranking fitting solution must be reached, or a plateau must be reached such that successive iterations no longer produce better results.

IV. GAs Tests

Several experiments have been performed to validate the algorithms presented in the previous section.

The parameter optimization GAs were tested with a microrobot composed of several rotation modules performing a snake-like motion. The locomotion parameters to be optimized by the algorithm were amplitude, angular velocity and phase. These parameters correspond to the sinusoidal movement that the robot is performing. Three different scenarios were chosen: open air, inside a pipe, and undulating terrain. The configuration parameters of each experiment are shown in Table IV.

For the configuration demand GA, a microrobot composed of several touch, rotation and helicoidal modules was placed inside a pipe with an elbow. The configuration parameters of this experiment are shown in Table V.

TABLE IV
GA Experiments: Parameter Optimization

Name	Exp 1	Exp 2	Exp 3	Exp 4
Population	40			
Gene type	Bit			
Chromosome	21 bits: Amplitude + Angular Velocity + Phase			
Generations	175	315	100	150
Crossover prob	0.7	0.8	0.7	0.75
Mutation prob	0.005	0.01	0.02	0.001
Fitness function	Distance covered			Distance covered + consump.
Terrain	Open air	Straight pipe: 36 mm ⌀	Undulating terrain	Open air

TABLE V
GA Configuration Demand Experiment: Elbow Negotiation

Name	Value
Population	40
Gene type	Module ID
Chromosome	8 genes (module ID)
Generations	25
Crossover prob	0.75
Mutation prob	0.005
Fitness function	Distance covered
Terrain	38 mm diameter straight pipe with an elbow

A. Parameter Optimization in Open Air (Exp 1)

In this experiment, the microrobot composed of six rotation modules is performing a snake-like motion. The results of the algorithm can be observed in Fig. 8. From a population of 40 randomly selected chromosomes, the best individual was obtained in generation 172: "11110011111111010000," which corresponds to the values: 85° amplitude, 15 rad/s angular velocity, and 1.97 rad phase.

It is notable that the best individual has the highest possible values for angular velocity and amplitude. This finding is reasonable because high angular velocity and amplitude values correspond to high wave propagation velocity and thus to more rapid robot motion.

B. Parameter Optimization in a 36 mm ø Pipe (Exp 2)

In this experiment, the microrobot composed of six rotation modules is performing a snake-like movement inside a pipe that is 36 mm in diameter (Fig. 10). The results of the algorithm (whose parameters can be observed in Table IV) are shown in Fig. 9. From a population of 40 randomly selected chromosomes, the best individual was obtained in generation 262: "01001011111100010000," which corresponds to the values: 26.2° amplitude, 14 rad/s angular velocity, and 0.39 rad phase.

The best individual maintains a high value for angular velocity, but the amplitude is smaller than that in the previous example. This finding is reasonable because inside the pipe, high amplitude sinusoidal waves are impossible. As in the previous experiment, higher angular velocities caused greater wave propagation velocities.

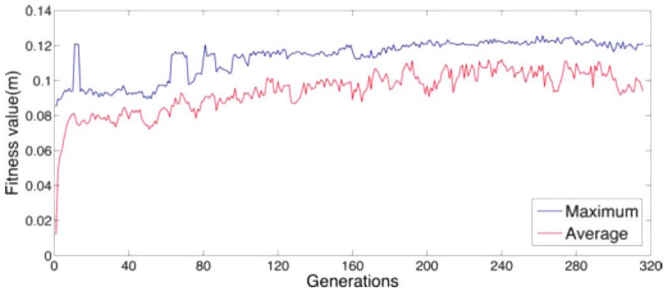

Fig. 9. Results of the GA inside a pipe (parameter optimization).

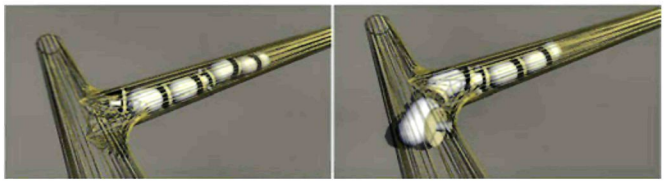

Fig. 10. GA in a 36-mm pipe.

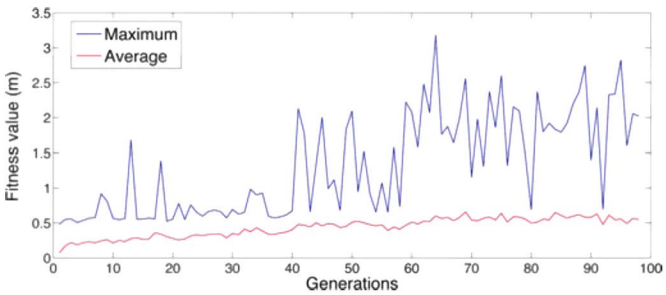

Fig. 11. Results of the GA in undulating terrain (parameter optimization).

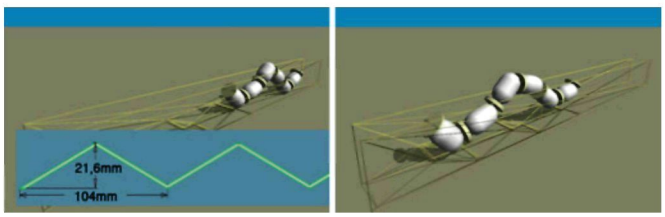

Fig. 12. GA experiment using undulating terrain.

C. Parameter Optimization in an Undulating Terrain (Exp 3)

In this experiment, a microrobot composed of eight rotation modules is performing a snake-like motion over undulating terrain as shown in Fig. 12. The results of the algorithm (whose parameters can be observed in Table IV) are shown in Fig. 11. The role of the walls is to prevent the microrobot from falling to the side.

From a population of randomly selected 40 chromosomes, the best individual was obtained in generation 63: "11001011111000011011 1," corresponding to the values: 71° amplitude, 14 rad/sangular velocity and 1.36 rad phase.

The results are different from those of previous experiments, showing an adjustment to the undulating terrain. The new amplitude and angular velocity are related to the peaks and valleys of the undulating terrain.

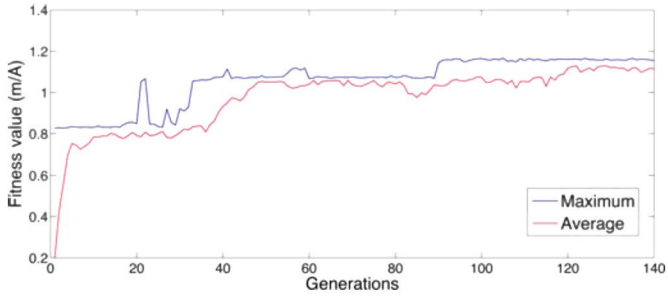

Fig. 13. Results of the GA in open air considering consumption (parameter optimization).

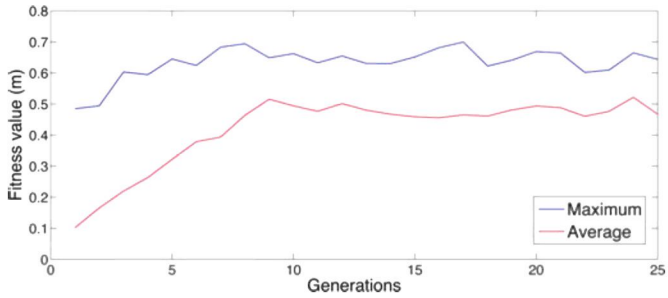

Fig. 14. Results of the GA in a pipe with an elbow (configuration demand).

D. Parameter Optimization of Power Consumption in Open Air (Exp 4)

In these experiments, a microrobot composed of eight rotation modules is performing a snake-like movement on flat terrain in the open air.

The fitness function has been modified and represents not only the distance covered but also the estimated power consumption of the microrobot. In this way, the test rewards the fastest microrobot with the lowest power consumption.

The fitness function used is represented by the following equation:

$$\text{Value}_{\text{fitness function}} = \frac{1000 \cdot \text{distance}}{50 + \text{consumption}}. \quad (1)$$

The results of the algorithm (whose parameters can be observed in Table IV) are shown in Fig. 13.

From a population of 40 randomly selected chromosomes, the best individual was obtained in generation 143: "11101011111010111010001," which corresponds to the values: 82° amplitude, 14.5 rad/s angular velocity, and 2 rad phase.

The angular velocity and amplitude obtained are slightly smaller than those obtained without consumption optimization, which seems logical. The consumption value for this individual is 690 mA. The consumption values vary from 0 to 2038 mA within the population. The importance of consumption can be increased by adjusting the parameters of (1).

E. Configuration Demand in a Pipe Elbow

This experiment was designed to determine the optimum configuration of a robot composed of 1 touch module and 8 rotation or helicoidal modules. The touch module should be in the

first position, and the other 8 modules can be either rotation or helicoidal.

From a population of 40 randomly selected individuals, the best individual was obtained in generation 17: "10101011" corresponding to the values: "THRHRHRHH," where "T" stands for touch, "H" for helicoidal, and "R" for rotation.

The results of the algorithm are shown in Fig. 14. The best configuration is the one with the maximum number of helicoidal modules that is able to negotiate the elbow. The helicoidal modules must be placed between the rotation modules so that they can turn. It is possible to position two consecutive helicoidal modules at the end of the microrobot because they can turn with the previous rotation module.

V. Conclusion and Future Work

This paper has presented a GA-based optimization procedure for bioinspired heterogeneous modular multiconfigurable chained microrobots. The integration of a simulator into the control architecture makes it possible to optimise the behavior of the robot in very different environments and thus to develop locomotion patterns very rapidly.

The offline control algorithms have been validated by optimizing the locomotion parameters in several experiments, including tests in the open air, in pipes, and on undulating terrain. These experiments have also made it possible to optimize the modular configuration of the robot composed of five heterogeneous modules. The use of offline GAs has proven to be a way to optimize the configuration of heterogeneous modular robots and a way to develop new gaits. Although offline control has been previously used in homogenous modular robots, this is the first time that it has been used in heterogeneous modular robots.

Future work will focus on the online use of the optimization algorithms, especially to make reconfiguration adjustments mainly due to execution errors. Therefore, the robot could fix bugs such as engine breaks, or loss of surface adhesion.

References

[1] H. Wei, Y. Chen, J. Tan, and T. Wang, "Sambot: A self-assembly modular robot system," *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 4, pp. 745–757, Aug. 2011.

[2] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh, "Multimode locomotion via superbot reconfigurable robots," *Auton. Robots*, vol. 20, no. 2, pp. 165–177, 2006.

[3] H. Kurokawa, K. Tomita, A. Kamimura, E. Yoshida, S. Kokaji, and S. Murata, "Distributed self-reconfiguration control of modular robot m-tran," in *Proc. IEEE Int. Conf. Mechatron. Autom.*, Jul. 2005, vol. 1, pp. 254–259.

[4] M. H. Yim, Y. Zhang, K. D. Roufas, D. G. Duff, and C. Eldershaw, "Connecting and disconnecting for chain self-reconfiguration with polybot," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 4, pp. 442–451, Dec. 2002.

[5] A. Castano, A. Behar, and P. Will, "The conro modules for reconfigurable robots," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 4, pp. 403–409, Dec. 2002.

[6] M. Jorgensen, E. Ostergaard, and H. Lund, "Modular atron: Modules for a self-reconfigurable robot," presented at the 2004 IEEE/RSJ Int. Conf. Intell. Robots Syst., Sendai, Japan, 2004.

[7] M. Yim, "New locomotion gaits," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1994, pp. 2508–2514.

[8] K. Stoy, W.-M. Shen, and P. Will, "Using role-based control to produce locomotion in chain-type self-reconfigurable robots," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 4, pp. 410–417, Dec. 2002.

[9] M. Sato, M. Fukaya, and T. Iwasaki, "Serpentine locomotion with robotic snakes," *IEEE Control Syst. Mag.*, vol. 22, no. 1, pp. 64–81, Feb. 2002.

[10] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic locomotion design and experiments for a modular robotic system," *IEEE/ASME Trans. Mechatronics*, vol. 10, no. 3, pp. 314–325, Jun. 2005.

[11] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 4, pp. 431–441, Dec. 2002.

[12] E. H. Ostergaard and H. H. Lund, "Evolving control for modular robotic units," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, 2003, vol. 2, pp. 886–892.

[13] D. Marbach and A. Ijspeert, "Online optimization of modular robot locomotion," in *Proc. IEEE Int. Conf. Mechatron. Autom.*, 2005, pp. 248–253.

[14] H. Hamann, J. Stradner, T. Schmickl, and K. Crailsheim, "A hormone-based controller for evolutionary multimodular robotics: From single modules to gait learning," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[15] B. Dong and Y. Li, "Multi-objective-based configuration generation and optimization for reconfigurable modular robot," in *Proc. Int. Conf. Inf. Sci. Technol.*, Mar. 2011, pp. 1006–1010.

[16] K. Baizid, R. Chellali, A. Yousnadj, A. Meddahi, and T. Bentaleb, "Genetic algorithms based method for time optimization in robotized site," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 1359–1364.

[17] J. Abbott, Z. Nagy, F. Beyeler, and B. Nelson, "Robotics in the small—Part I: Microbotics," *IEEE Robot. Autom. Mag.*, vol. 14, no. 2, pp. 92–103, Jun. 2007.

[18] P. Jantapremjit and D. Austin, "Design of a modular self-reconfigurable robot," in *Proc. Australian Conf. Robot. Autom.*, 2001, pp. 38–43.

[19] A. Brunete, J. Torres, M. Hernando, and E. Gambao, "Heterogenous multicofigurable chained micro-robot for exploration of small cavities," *Autom. Construct. Mag.*, vol. 21, pp. 184–198, 2011.

[20] A. Brunete, J. Torres, M. Hernando, and E. Gambao, "A proposal for a multi-drive heterogeneous modular pipe-inspection micro-robots," *Int. J. Inf. Acquisit.*, vol. 5, pp. 111–126, 2008.